CLAIMS

1. A method for testing a voice algorithm module for compliancy in a voice framework, the method comprising the steps of:

filling a scratch memory location with a predetermined fill pattern associated with a function to be tested,

saving context information to a predetermined memory location, wherein the context information is specific to the function and is to be preserved for a call to the function wherein the context information comprises at least one variable required for a subsequent call to the function;

configuring a timer interrupt to interrupt the function after each cycle to minimize pipeline conflicts;

calling the function;

verifying context information by comparing a current version of the context information with the saved version of the context information; and

writing an error code to an output file based on the verification of context information.

2. The method of claim 1, wherein context information comprises at least one of status registers, general purpose registers and stack pointers.

3. The method of claim 2 further comprising the step of placing a predetermined number of patterns in a predetermined number of adjacent words on the stack; after the call to the function, checking that the predetermined words have a same relative address to top of the stack.

4. The method of claim 1, wherein the step of verifying context information further comprises the step of verifying status registers by comparing status registers with saved context information.

5. The method of claim 1, wherein the step of verifying context information further comprises the step of verifying general purpose registers by comparing general purpose registers with saved context information.

6. The method of claim 1, wherein the step of verifying context information further comprises the step of verifying allocated scratch space was used by comparing the scratch space and the fill pattern.

7. The method of claim 1, wherein the fill pattern is a 16-bit pattern.

8. The method of claim 1, wherein pipeline conflicts arises when executing a plurality of instructions in an assembly line manner.

9. The method of claim 1, further comprising the step of performing memory test initialization steps.

10. The method of claim 1, further comprising the steps of:

saving a return value associated with the function; and

comparing the return value with a plurality of known values of a maintained data file that represents correct values of the function called.

11. The method of claim 1, wherein the step of configuring a timer interrupt further comprises the steps of:

initializing a timer in a time control register;

updating a program counter of an interrupted line of a code, for comparison with a previous value of the program counter; and

generating an interrupt for each line of code of the function based on an expiration of the timer.

12. A method for testing a voice algorithm module for compliancy in a voice framework, the method comprising the steps of:

initializing a timer in a time control register;

saving context information associated with a function to a predetermined context memory location;

updating a program counter of an interrupted line of a code, for comparison with a previous value of the program counter;

generating an interrupt for each line of code of the function based on an expiration of the timer; and

using a stack pointer value to fill a scratch memory location with a fill pattern, associated with a called function.

13. The method of claim 12, further comprising the step of updating at least one counter to account for at least one extra clock cycle.

14. The method of claim 13, wherein the counter indicates the number of times an instruction has been at a particular program counter address.

15. The method of claim 12, further comprising the step of clearing any pending timer interrupts.

16. The method of claim 12, further comprising the step of restoring the stack pointer value.

17. The method of claim 12, further comprising the step of setting the timer to enable a next interrupt to occur after a next instruction is processed for comprehensive test coverage.

18. The method of claim 12, wherein the context information comprises at least one variable that is required for a subsequent call to the function.

19. The method of claim 12, further comprising the step of:

determining a time interrupt, wherein if a time interrupt is detected, a period register of the program counter is decremented, else the period register of the program counter is initialized to a maximum 16-bit value.

20. The method of claim 12, wherein the context information comprises at least one of status registers, general purpose registers and stack pointers.

21. A method for testing a voice algorithm module for compliancy in a voice framework, the method comprising the steps of:

verifying an interrupt service routine is running;

saving a value of a timer register as a temporary value;

saving context information associated with a function to a predetermined context memory location;

updating a program counter of an interrupted line, for comparison with a previous value of the program counter;

using a stack pointer to fill a scratch memory location with a fill pattern, associated with a called function; and

subtracting the temporary value of the timer register from an initial time value to determine a latency value.

22. The method of claim 21, further comprising the step of clearing any pending timer interrupts.

23. The method of claim 21, further comprising the step of restoring the stack pointer value.

24. The method of claim 21, further comprising the step of setting the timer to enable a next interrupt to occur after a next instruction is processed for comprehensive test coverage.

25. The method of claim 21, wherein hardware supporting the voice algorithm comprises at least two timers.

26. The method of claim 21, wherein the latency value is used to determine the duration of an instruction.

27. The method of claim 21, wherein a maximum latency value is set to 66 cycles.

28. The method of claim 21, wherein a maximum latency value is set to 42 cycles.

29. The method of claim 21, wherein context information comprises at least one variable that is required for a subsequent call to the function.